Asian Journal of Information Technology 8 (1): 1-7, 2009
ISSN: 1682-3915

Hybrid Neural Network-Monte Carlo Simulation for Stock Price Index Prediction

Joko Lianto Buliali, Chastine Fatichah and Mudji Susanto
Department of Informatics, Faculty of Information Technology,
Sepuluh Nopember Institute of Technology (ITS), Kampus ITS Keputih,
Sukolilo, Surabaya-60111, Indonesia

Abstract: This research uses Monte Carlo simulation to increase the accuracy of neural network prediction on a limited number of composite stock price index. The case study is Indonesian composite stock price index (i.e., Jakarta Composite Index (JCI)) from July 1997 to December 2007. Monte Carlo simulation is used to generate additional data from the available data, which is then fed into neural network to forecast future data. Testing results show that the output of hybrid neural network-Monte Carlo simulation system produces significantly lower Mean Absolute Percentage Error (MAPE) than the output of neural network without data from Monte Carlo simulation.

Key words: Monte Carlo simulation, neural network, composite stock price index, Jakarta composite index, prediction

INTRODUCTION

Financial forecasting (including security returns, stock price, foreign exchange rates, capital flow, etc.) has been a focus of research for the last few years and various techniques have been proposed. Below is a number of such researches and the techniques used:

- Non-linear modeling, where financial data are regarded as non-linear and therefore require non-linear modeling (Gradojevic, 2006)
- Fuzzy rule based system, where the relationship among factors are modeled in a fuzzy relationship (Chang and Liu, 2006)
- Neural networks, which analyze relationships among complex financial data and store relationships in terms of weights as a result from training (Yao and Tan, 2001; Zhang, 2004; O'Connor and Madden, 2006; Kim, 2006; Tsang *et al*., 2007)

Neural networks gain popularity among other techniques for financial forecasting due to the relatively easy implementation and training. Complex relationship among data are tuned info the network during training. The more (and accurate) training data, the more accurate the network will perform. Unfortunately abundant data is not always available. In cases, where there are only limited number of data, neural networks can perform rather satisfactorily only when predicting short periods. The level of accuracy in such case is not high. If long period prediction is forced, the error will be excessively large.

The hypothesis in this research is that the use of Monte Carlo simulation for generating additional data can increase the accuracy of prediction of neural networks in situations where only limited number of composite stock price index data are available. Here Monte Carlo is used to increase the number of data according to the designated distribution of the sample data. As the system is intended to forecast composite stock price index, Exponential Generalized Autoregressive Conditional Heteroskedasticity (EGARCH) model is used which is the most common model used in composite stock price index forecast.

MATERIALS AND METHODS

As mentioned in the selection of input variables is fundamental to accurately forecast the stock movements (Coupelon, 2007). It primarily depends on a clear understanding of the economical background of the stock price to forecast.

In this research, the inputs chosen are Exponential Moving Average (EMA), Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD) and Stochastic Oscillator (SO). These technical analysis indicators are commonly used as input for neural network.

Corresponding Author: Joko Lianto Buliali, Department of Informatics, Faculty of Information Technology, Sepuluh Nopember Institute of Technology (ITS), Kampus ITS Keputih, Sukolilo, Surabaya-60111, Indonesia

Moving Average (MA): Moving average is suitable for identifying whether a security is moving in an uptrend or a downtrend depending on the direction of the moving average. The three most common types of moving averages are simple, linear and exponential (Janssen *et al*., 2009). Exponential Moving Average (EMA) is more responsive to new information relative to the simple moving average. As responsiveness is one of the key factors in calculating trend. EMA is the one used in this research. Equation 1 shows the formula of EMA (Colby and Thomas, 2002):

$$EMA_t = EMA_{t-1} + S_f(P_t - EMA_{t-1}) \quad (1)$$

where:

EMA_t = Exponential moving average
EMA_{t-1} = Previous moving average value
S_t = Smoothing factor = 2/(n+1)
n = Number of periods

Relative Strength Index (RSI): RSI is popular as it can be interpreted easily. The RSI compares the magnitude of a stock's recent gains to the magnitude of its recent losses and turns that information into a number that ranges from 0-100 (StockCharts, 2009). Equation 2 shows the formula of RSI (Colby and Thomas, 2002):

$$RSI = 100 - \left(\frac{100}{1+RS}\right) \quad (2)$$

where, RS is the ratio of the exponential moving average of 14-day gains divided by the absolute value of the exponential moving average of 14-day losses.

Moving Average Convergence Divergence (MACD): MACD is composed of two lines. The first line is the difference between two exponential moving averages (i.e., a 26 and 12-days moving average of daily close price). The second is a Signal line which is the EMA of the first line (i.e., a 9-days EMA). MACD line will oscillate along the time axis to show buy/sell opportunities. (Magdefrau, 2006). Equation 3 shows the formula of MACD (Colby and Thomas, 2002):

$$\begin{aligned} MACD1 &= EMA_{12} - EMA_{26} \\ Signal &= EMA_9 \\ MACD &= MACD1 - signal \end{aligned} \quad (3)$$

where:

EMA_{12} = 12-period EMA
EMA_{26} = 26-period EMA
EMA_9 = 9-period EMA

Stochastic Oscillator (SO): SO is a technical momentum indicator that compares a security's closing price to its price range over a given time period. The oscillator's sensitivity to market movements can be reduced by adjusting the time period or by taking a moving average of the result. This indicator is calculated with the formula shown in Eq. 4 (Investopedia, 2009):

$$K(\%) = \frac{CP_t - L_n}{H_n - L_n} \times 100 \quad (4)$$

where:

K = Stochastic oscillator
CP_t = The latest closing price of the stock or contract
L_n = n-period low price of the stock or contract
H_n = n-period high price of the stock or contract
n = Any number (a range of 5-21 is recommended)

Figure 1 shows the four technical indicators chosen as input data, both for neural network and hybrid neural network-Monte Carlo simulation system.

Before fed into neural network, data is preprocessed to facilitate the learning process of neural network. Data is normalized into 0 and +1 scale. Linear scaling method is applied to keep the uniformity of the distribution (Coupelon, 2007).

In designing a neural network system for stock price index prediction, several points must be considered (Coupelon, 2007; Azoff , 1994; Zhang, 2004):

Number of input neurons: As there are four inputs (EMA, RSI, MACD and SO), four input neurons for the neural network.

Number of hidden layers: In this research one hidden layer is used.

Number of hidden neurons: This number is determined during experimentation (training phase of the neural network development). If several experimentations yield different number of hidden neurons with more or less the same error, the one with the least number of hidden neurons is chosen.

Number of output neurons: As there is only one output is required (stock price index prediction), one output neuron is used.

Transfer functions: In this research sigmoid transfer function is used, which is the most commonly used.

Figure 2 shows the model of Neural Network for forecast composite stock price index.

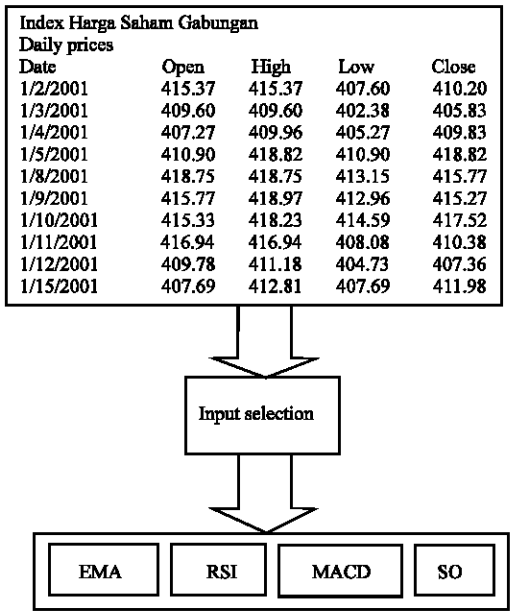

Fig. 1: Input variables

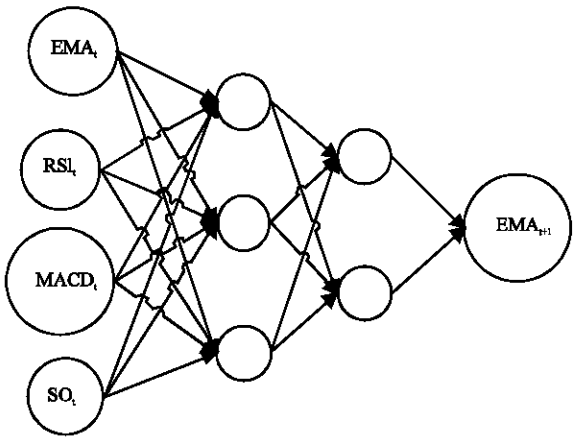

Fig. 2: Neural network model

Neural network training is an iterative process so that the network tunes complex relationship among data into neuron weight and therefore can be used for prediction future data. The goal is to minimize errors, which indicates that the model behaves as closely as possible as the index behaviour in the real world. Error function used to evaluate the model is MSE (the mean squared of the difference of the output of the network and the targeted output) which is the commonly used error function. This error function is used to update the weights in the model so that the output is adjusted as the data (Back Propagation Networks (BPN)). The algorithm used for the training is as follows (Bourg and Seeman, 2004):

1. Start training with a group of input data and targeted output
2. Initialize all weights in the network with some small random numbers
3. Input each input data from the group and calculate the output
4. Compare the output of the network with targeted output and calculate the error
5. Change weights in the network to decrease error and repeat the process

Steps 4 (error calculation) and 5 (weight change) constitute the important part of back propagation algorithm. This error calculation (MSE calculation) cannot only to determine whether the output is correct or not, but also to determine the degree of error of the output. The main goal is the get the least MSE by changing the weights of the neurons iteratively. The formula is as shown in Eq. 5 (Arbib, 2002):

$$\delta_i^0 = \Delta n_i^o \times f'\left(n_{ci}^o\right) \quad (5)$$

where:

δ_i^o = The error of ith neuron output
Δn_i^o = The difference of targeted output and the ith neuron output
$f'(n_{ci}^o)$ = The derivative of activation function (sigmoid function) for the ith neuron output by using the derivative of sigmoid function

Eq. 5 becomes Eq. 6 (Arbib, 2002):

$$\delta_i^o = \left(n_{di}^o - n_{ci}^o\right) \times n_{ci}^o \times \left(1 - n_{ci}^o\right) \quad (6)$$

where:

n_{di}^o = The value of ith targeted output neuron
n_{ci}^o = The value of ith output of the output layer

Error calculation in hidden layer can be calculated by using formula in Eq. 7 (Arbib, 2002):

$$\delta_i^h = \left(\Sigma \delta_j^o \times w_{ij}\right) \times f'\left(n_{ci}^h\right) \quad (7)$$

where, n_{ci}^h is the value of output neuron of the hidden layer. It is seen that the error from every neuron in hidden layer is an error function associated with every neuron in output layer, where the error of neuron output is multiplicated with the weight of the neuron. This implies that the error calculation changes the weights of the neurons from output layer towards input layer. By substituting activation function to Eq. 7 and 8 is obtained.

$$\delta_i^h = \left(\Sigma\delta_j^o \times w_{ij}\right) \times n_{ai}^h \times \left(1 - n_{ai}^h\right) \quad (8)$$

Error calculation for input layer is not required as the value of each neuron in input layer is known. By using the above error calculation, weight change can be calculated. Equation 9 shows the formula for weight change:

$$\Delta w = \eta \times \delta_i \times n_i \quad (9)$$

where:

η = Learning rate
δ_i = The error of the neuron whose weight will be changed
n_i = The value of the neuron whose weight will be changed

The new weight is obtained by adding the old weight and Δw. Weight change is carried out on every weight and the change value (Δw) will be different on every weight. Learning rate is a multiplier that affects the impact of every weight change. In this research the learning rate is set between 0.25 and 0.5. If the value is too large, the weights will oscillate. If the value is too small, the training will take a long time to settle.

To achieve an even shorter training time, momentum is used in the weight change process. The formula to calculate weight change that incorporates momentum is shown in Eq. 10 (Katagiri, 2000).

$$\Delta w = \eta \times \delta_i \times n_i + \alpha(\Delta w') \quad (10)$$

where:

$\Delta w'$ = The weight change from previous iteration
α = The momentum factor which ranges from 0.0-1.0

Another important part of the proposed system is Monte Carlo simulation. A Monte Carlo experiment involves the following steps (Schmidheiny, 2007):

1. Draw a (pseudo) random sample of size N for the stochastic elements of the stochastic model from their respective probability distribution functions
2. Assume values for the exogenous parts of the model or draw them from their respective distribution function
3. Calculate the endogenous parts of the statistical model
4. Calculate the value of interest (e.g., the estimate)
5. Replicate step 1-4 R times
5. Examine the empirical distribution of the R-values

In Monte Carlo experiments, random number generator produces a sequence of numbers from a mathematical algorithm, which produces a sequence of pseudo random numbers that are identically and independently distributed. The random numbers are called pseudo random numbers as they are not random as the algorithm describes the purely deterministic relationship between the numbers. However, with a good generator, the numbers are indistinguishable from sequences of genuinely random numbers and pass usual statistical tests of independence (Schmidheiny, 2007).

Input data is preprocessed before being fed into neural network. Preprocessing is required to change input data into matrix form to be further processed by MatLab software. Data from preprocessing constitute 50% of the data for neural network as shown in Fig. 3. The remaining 50% data is fed from Monte Carlo simulation.

Financial model for Monte Carlo simulation is Exponential Generalized Autoregressive Conditional Heteroskedasticity (EGARCH). In this research Gaussian data distribution is used for the GARCH model. Figure 4 shows Monte Carlo model for generating the remaining 50% data.

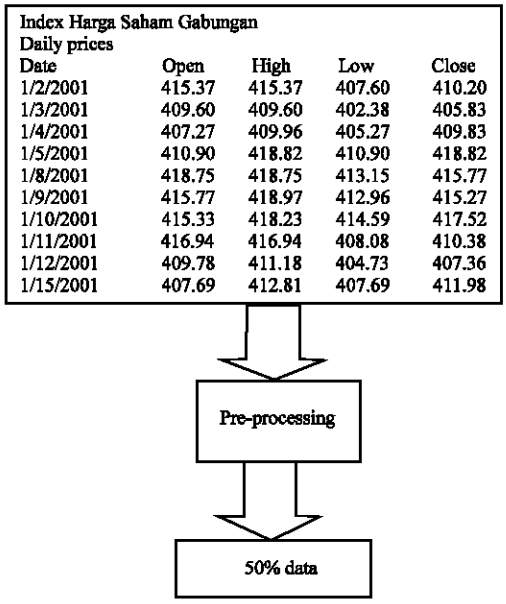

Fig. 3: Data preprocessing

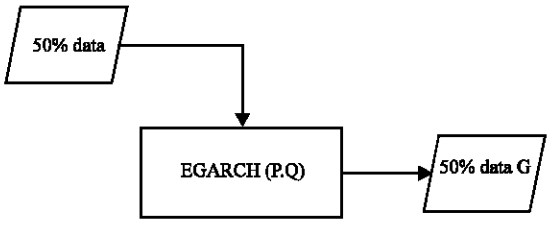

Fig. 4: Monte Carlo model

RESULTS AND DISCUSSION

Testing was carried out by comparing the Mean Absolute Percentage Error (MAPE) calculation for the output of neural network system and that of hybrid neural network-Monte Carlo simulation system. Data for testing was Jakarta Composite Index (JCI)) from July 1997 to December 2007. The formula for MAPE calculation is shown in Eq. 11:

$$MAPE = \left(\frac{1}{m} \sum \frac{|n_c - n_d|}{n_c} \right) \times 100\% \quad (11)$$

where:

n_c = The output value of neural network
n_d = The targeted output
m = The number of outputs of the neural network

Two kinds of training were carried out, one training with 6-months data (experiment 1), another with 12-months data (experiment 2). This is meant to see the effects of the number of training data to system output. The length of prediction was varied from 3 months up to 9 months. This is meant to see the accuracy of the prediction for different time horizon. After training, both systems were ready for evaluation.

Experiment 1: The first step to evaluate the system is to calculate for the output of neural network system and that of hybrid neural network-Monte Carlo simulation system. Table 1 shows the results obtained.

The Monte Carlo system is run for ten replications initially. Next, the required (minimum) number of replication for each experiment is calculated for each category using in Eq. 12:

Table 1: MAPE of output of NN system and hybrid NN-MC simulation system

	Experiment 1					
	3-months MAPE		6-months MAPE		9-months MAPE	
Training	NN	NN + MC	NN	NN + MC	NN	NN + MC
Jul 97-Dec 97	0.396760	0.072849	0.123120	0.149650	0.187370	0.134120
Oct 97-Mar 98	0.192120	0.016999	0.202310	0.033667	0.251050	0.057965
Jan 98-Jun 98	0.231170	0.098474	0.102580	0.221690	0.145590	0.064474
Apr 98-Sep 98	0.224560	0.090066	0.256300	0.027950	0.276240	0.044980
Jul 98-Dec 98	0.209980	0.129430	0.304280	0.039805	0.229270	0.140060
Oct 98-Mar 99	0.008514	0.127410	0.060011	0.013791	0.023472	0.032656
Jan 99-Jun 99	0.100940	0.023224	0.219640	0.153930	0.165080	0.012806
Apr 99-Sep 99	0.135470	0.049531	0.152410	0.117840	0.162870	0.147160
Jul 99-Dec 99	0.292650	0.080310	0.202440	0.088502	0.178850	0.058851
Oct 99-Mar 00	0.087401	0.028043	0.024292	0.024287	0.028083	0.047190
Jan 00-Jun 00	0.057708	0.101290	0.068052	0.205380	0.194670	0.077743
Apr 00-Sep 00	0.211090	0.012041	0.174850	0.026419	0.050402	0.010790
Jul 00-Dec 00	0.113660	0.292480	0.126510	0.181890	0.130060	0.124160
Oct 00-Mar 01	0.433820	0.189580	0.375110	0.248920	0.430740	0.295010
Apr 01-Sep 01	0.226490	0.090602	0.380110	0.183110	0.202060	0.219040
Jul 01-Dec 01	0.361880	0.384950	0.141890	0.306790	0.345460	0.132600
Oct 01-Mar 02	0.011305	0.093557	0.191980	0.000161	0.322240	0.190200
Jan 02-Jun 02	0.158000	0.035704	0.207180	0.046327	0.145930	0.045199
Apr 02-Sep 02	0.067722	0.040419	0.153290	0.109510	0.177060	0.035572
Jul 02-Dec 02	0.316570	0.174790	0.198240	0.245780	0.172220	0.140500
Oct 02-Mar 03	0.124190	0.038334	0.022821	0.121760	0.019288	0.063131
Jan 03-Jun 03	0.221050	0.098127	0.040785	0.126530	0.065457	0.073505
Apr 03-Sep 03	0.083059	0.102140	0.229490	0.104360	0.081172	0.121500
Jul 03-Dec 03	0.050633	0.066768	0.130160	0.037059	0.112610	0.113150
Oct 03-Mar 04	0.125600	0.158870	0.077018	0.087980	0.079999	0.070863
Jan 04-Jun 04	0.207200	0.019031	0.077452	0.053768	0.049335	0.052762
Apr 04-Sep 04	0.002233	0.022397	0.050838	0.025010	0.040283	0.004066
Jul 04-Dec 04	0.033208	0.044113	0.014926	0.057000	0.048789	0.019452
Oct 04-Mar 05	0.117390	0.059449	0.143590	0.045808	0.049544	0.044724
Jan 05-Jun 05	0.172440	0.075549	0.192680	0.074961	0.116890	0.041566
Apr 05-Sep 05	0.151770	0.096428	0.111400	0.035335	0.046698	0.086379
Jul 05-Dec 05	0.021223	0.007506	0.043343	0.071960	0.017597	0.049164
Oct 05-Mar 06	0.061403	0.027307	0.063240	0.035534	0.020274	0.031496
Jan 06-Jun 06	0.125120	0.040187	0.039036	0.065687	0.020817	0.031404
Apr 06-Sep 06	0.036041	0.043295	0.026404	0.043134	0.030466	0.014518
Jul 06-Dec 06	0.063903	0.010318	0.032375	0.012246	0.0448	0.021805
Oct 06-Mar 07	0.0079888	0.016333	0.024012	0.0027511	0.020085	0.0007578
Average	0.147088	0.082646	0.134707	0.092602	0.126563	0.077063
SD	0.1118387	0.0784639	0.0965579	0.0787785	0.1042695	0.0647238
Max.	0.433820	0.384950	0.380110	0.306790	0.430740	0.295010
Min.	0.002233	0.007506	0.014926	0.000161	0.017597	0.000758

$$R = \left(t_{R-1,\alpha/2} \times \frac{SD}{\varepsilon} \right)^2 \quad (12)$$

where:

R = The required number of replication
α = The critical value (set to 0.05)
R-1 = The degree of freedom = (10 - 1) = 9
SD = The standard deviation from the experiments
ε = Maximum tolerable error (set as 10% from the associated MAPE)

Table 2 shows the number of replication for each experiment.

Finally statistical t-test was carried out with null Hypothesis (H_0) that the MAPE of the output of neural network system does not differ from that of hybrid neural network-Monte Carlo simulation system. Each category was evaluated after the required number of replication was carried out. The alpha in the t-test is 0.05. The values are shown in Table 3.

As the critical value of t-test is 1.67, H_0 will be rejected for areas where t = 1.67. The value of t for 3-months forecast period is 2.869, therefore, H_0 is rejected (shich means that the MAPE of the output of neural network system differs from that of hybrid neural network-Monte Carlo simulation system). Similarly, H_0 is rejected for 6-months forecast period and for 9-months forecast period. It is also seen that the MAPE of hybrid neural network-Monte Carlo simulation system is lower than that of neural network by itself.

Experiment 2: The steps in this experiment is the same as the steps in experiment 1. The difference is only in the length of data for training. While in experiment 1, 6-months data is used in experiment 2, 12-months data are used instead. Although, replication calculation reveals that different number of replication is required in each category, the statistical t-test yields the same conclusion of rejecting H_0 and the MAPE of hybrid neural network-Monte Carlo simulation system is lower than that of neural network by itself.

The general conclusion from the two experiments is as follows. As H_0 is rejected in all cases, it can be concluded that hybrid neural network-Monte Carlo simulation system performs significantly different from neural network by itself (without data from Monte Carlo simulation). Since the MAPE of hybrid neural network Monte Carlo simulation system is lower than that of neural network by itself, it is concluded that hybrid neural network-Monte Carlo simulation system performs better than neural network by itself.

Table 2: Number of replication required for each experiment

	No. replication required					
	3-months		6-months		9-months	
Training	NN	NN + MC	NN	NN + MC	NN	NN + MC
Jul 97-Dec 97	1	2	1	1	1	1
Oct 97-Mar 98	1	6	1	4	1	2
Jan 98-Jun 98	1	1	1	1	2	1
Apr 98-Sep 98	2	5	1	4	1	2
Jul 98-Dec 98	1	1	1	2	1	1
Oct 98-Mar 99	4	5	4	4	5	3
Jan 99-Jun 99	2	2	1	4	1	5
Apr 99-Sep 99	1	1	1	1	1	1
Jul 99-Dec 99	1	3	1	2	1	1
Oct 99-Mar 00	3	2	3	3	3	2
Jan 00-Jun 00	4	2	2	1	2	2
Apr 00-Sep 00	1	13	2	4	2	13
Jul 00-Dec 00	1	2	1	1	1	1
Oct 00-Mar 01	1	2	1	1	1	1
Apr 01-Sep 01	1	2	1	1	1	1
Jul 01-Dec 01	1	2	1	1	2	1
Oct 01-Mar 02	4	6	1	9	1	1
Jan 02-Jun 02	1	4	2	3	1	2
Apr 02-Sep 02	2	2	2	2	1	1
Jul 02-Dec 02	1	5	1	1	1	1
Oct 02-Mar 03	1	1	7	1	2	1
Jan 03-Jun 03	1	1	1	1	2	1
Apr 03-Sep 03	1	1	2	1	1	1
Jul 03-Dec 03	1	1	3	4	1	2
Oct 03-Mar 04	1	2	1	1	1	1
Jan 04-Jun 04	1	5	1	1	1	1
Apr 04-Sep 04	9	2	3	2	5	6
Jul 04-Dec 04	2	7	3	3	2	4
Oct 04-Mar 05	1	3	1	2	1	2
Jan 05-Jun 05	1	1	1	1	1	1
Apr 05-Sep 05	1	2	1	2	1	1
Jul 05-Dec 05	3	4	2	1	3	1
Oct 05-Mar 06	3	2	1	1	1	1
Jan 06-Jun 06	1	2	1	1	3	1
Apr 06-Sep 06	3	1	3	2	3	2
Jul 06-Dec 06	2	2	2	1	1	3
Oct 06-Mar 07	7	16	2	4	3	11
Jul 02-Dec 02	1	5	1	1	1	1
Oct 02-Mar 03	1	1	7	1	2	1

Table 3: T-test of the results

t-test	3 bln	6 bln	9 bln
t Stat	2.869	2.055	2.453
t Crit 1-tail	1.667	1.666	1.666
t Crit 2-tail	1.993	1.993	1.993

CONCLUSION

The proposed hybrid neural network-Monte Carlo simulation system has proved to produce smaller error than neural network model by itself on the case study of Jakarta composite index. The underlying concept here is that Monte Carlo simulation can be utilized to generate additional data from limited sample data. However, Monte Carlo is a simulation method, experiments using hybrid neural network-Monte Carlo simulation system requires more replication that neural network model by itself, because this hybrid system uses generated data from actual data.

REFERENCES

Arbib, M.A., 2002. The Handbook of Brain Theory and Neural Networks. 2nd Edn. The MIT Press, Cambridge, MA 02142, pp: 1-22. ISBN: 978-026201 1976.

Azoff, E.M., 1994. Neural Network Time Series Forecasting of Financial Markets. 1st Edn. John Wiley and Sons, Inc. New York, USA. ISBN: 978-0471943563.

Bourg, D.M. and G. Seeman, 2004. AI for Game Developers. 1st Edn. O'Reilly, Sebastopol, CA 95472. pp: 269-315. ISBN: 978-0596005559.

Chang, P.C. and C.H. Liu, 2006. A TSK type fuzzy rule based system for stock price prediction. Elsevier Expert Syst. Appl. J., 34 (1): 135-144. DOI: 10.1016/j.eswa.2006.08.020.

Colby, R.W. and A.M. Thomas, 2002. The Encyclopedia of Technical Market Indicators. 2nd Edn. McGraw-Hill, New York, pp: 1-44. ISBN: 978-1556230493.

Coupelon, O., 2007. Neural Network Modeling for Stock Movement Prediction: A State of the Art. http://olivier.coupelon.free.fr/Neural_network_modeling_for_stock_movement_prediction.pdf.

Gradojevic, N., 2006. Non-linear, Hybrid exchange rate modeling and trading profitability in the foreign exchange market. Elsevier J. Econ. Dynamics and Control, 31 (2): 557-574. DOI: 10.1016/j.jedc.2005.12.002.

Investopedia, 2009. Stochastic Oscillator. http://www.investopedia.com/terms/s/stochasticoscillator.asp.

Janssen, C., C. Langager and C. Murphy, 2009. Technical analysis: Moving Averages. http://www.investopedia.com/university/technical/techanalysis9.asp.

Katagiri, S., 2000. Handbook of Neural Networks for Speech Processing. 1st Edn. Artech House, Inc. Norwood, MA, pp: 1-1 to 1-30. ISBN: 978-0890069547.

Kim, K.J., 2006. Artificial neural networks with evolutionary instance selection for financial forecasting. Elsevier Expert Syst. Appl. J., 30 (3): 519-526. DOI: 10.1016/j.eswa.2005.10.007.

Magdefrau, C., 2006. Moving Average Convergence Divergence-MACD. www.stockval.com/papers/MACD.pdf.

O'Connor, N. and M.G. Madden, 2006. A neural network approach to predicting stock exchange movements using external factors. Elsevier Knowledge-Based Syst. J., 19 (5): 371-378. DOI: 10.1016/j.knosys.2005.11.015.

Schmidheiny, K., 2007. Monte Carlo experiment. Unversitat Pompeu Fabra. http://kurt.schmidheiny.name/teaching/montecarlo2up.pdf.

StockCharts, 2009. Relative Strength Index (RSI): Introduction. http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:relative_strength_index_rsi.

Tsang, P.M., P. Kwok, S. Choy, R. Kwan, S. Ng, J. Mak, J. Tsang, K. Koong and T.L. Wong, 2007. Design and implementation of NN5 for hong kong stock price forecasting. Elsevier Eng. Appl. Artificial Intell. J., 20 (4): 453-461. DOI: 10.1016/j.engappai.2006.10.002.

Yao, J. and C.L. Tan, 2001. Guidelines for financial forecasting with neural networks. http://www2.cs.uregina.ca/jtyao/Papers/guide iconip01.pdf.

Zhang, G.P., 2004. Business Forecasting with Artificial Neural Networks: An Overview. 1st Edn. Georgia State University, USA, pp: 1-22. ISBN: 978-15914 01766.